The Quality of Software Metrics

Dragoljub Pilipović, Dejan Simeunović
Faculty of Information Technology, Slobomir P University, Bijeljina
dragoljub.pilipovic@gmail.com, dejan.simeunovic@spu.ba

Critical Review
https://doi.org/ 10.7251/JIT2101061P UDC: 519.68:681.3.06]:336.71

Abstract: This paper discusses the definition, types, characteristic and construction of software metrics in the field of software development. Finally, an overview is given regarding the use of a software tool in software development in relation to software metrics in the field of banking.

Keywords: software engineering, software metrics, SEI, CISQ, ISO/IEC, banking.

Introduction

Nowadays, the discipline of software engineering is well-represented in practice as well as a scientific theory. The reason for this is the extensive use of software and, consequently, the need to develop the respective software. Despite the mass production of software, there is a lack of quality assurance of software and the process of its development. For the monitoring, control and prediction of the characteristics of software product, software metrics are often used [7] [8] [15]. Therefore, in this paper, in the context of software engineering, we define terms software quality and software metrics.

According to ISO organization [10], quality is defined towards object and speaks of the degree how inherent characteristics of the object fulfill required set of demands on it. The object represents any entity that can be seen or conceived, and its inherent characteristics are properties that exist in the object or its associated items. In the context of software engineering, the definition of quality of software as product quality, as well of software development as a quality development process can be applied.

This paper [16] shows that some software metric tools interpret and implement the definitions of software metrics in different ways. Therefore, any metric should be precisely defined prior to each application thereof. More specifically, the metric itself must be of good quality to have a product with high-quality software within itself. The metric must be defined first abstractly, second textually and with mathematical formulas and equations; and then, in the next step, in the form of algorithms and pseudo-code.

SOFTWARE QUALITY ASSURANCE

Among others, there are two important organizations in the world dealing with the concept of software quality and metrics:

1) CISQ (The Consortium for IT Software Quality), jointly organized by the Software Engineering Institute (SEI) at Carnegie Mellon University and the Object Management Group (OMG), and

2) ISO/IEC and their standard „ISO/IEC 9126 Software engineering — Product quality" and its successor ISO/IEC 25010:2011.

The last organization is more important and better known, therefore we will deal with their standards. The name of the standard is ISO/IEC 9126 Software engineering — Product quality. It was an international standard for the evaluation of software quality. The new standard with the same subject is ISO/IEC 25010:2011. The main goal of the ISO/IEC standard is to addresses some of the known human tendencies that can negatively affect at the delivery and perception of software development project. Therefore, the standard tries to develop a common understanding of the project's priorities, objectives

and goals. After clarifying, agreeing on the project priorities and subsequently converting abstract priorities to measurable values follow. The standard is divided into four parts:

- quality model (with name 9126-1),
- external metrics (9126-2),
- internal metrics (9126-3), and
- quality in use metrics (9126-4).[11][12][13][14]

The quality model presented in the first item of the standard, classifies software quality in a structured set of characteristics:

- Functionality, "a set of attributes that bear on the existence of a set of functions and their specified properties" (suitability, accuracy, interoperability, security, and functionality compliance).
- Reliability, "a set of attributes that bear on the capability of software to maintain its level of performance under stated conditions for a stated period of time" (maturity, fault tolerance, recoverability, reliability compliance).
- Usability, "a set of attributes that bear on the effort needed for use, and on the individual assessment of such use, by a stated or implied set of users" (understandability, learnability, operability, attractiveness, usability compliance).
- Efficiency, "a set of attributes that bear on the relationship between the level of performance of the software and the amount of resources used, under stated conditions" (time behaviour, resource utilization, efficiency compliance).
- Maintainability, "a set of attributes that bear on the effort needed to make specified modifications" (analyzability, changeability, stability, testability, maintainability compliance).
- Portability, "a set of attributes that bear on the ability of software to be transferred from one environment to another" (adaptability, installability, co-existence, replaceability, portability compliance).[11]

Each sub-characteristic (e.g. suitability) is further divided into attributes. An attribute is an entity which can be measured in the software product and links to software metrics.

SOFTWARE METRICS

In order to ensure quality of software it is necessary to measure multiple parameters. This should be done by defining software metrics, i.e. the type of measurement that is related to a software system, process or a related document. It is necessary to make a selection of parameters which will be measured and provide testing software products using the strategy of the necessary approaches to validation testing. All this is needed to get a software product that meets the requirements of customers, developed in accordance with the specification and which is free of errors.

In reference [9], we have given the first serious definition of software metrics (in fact, the definition defines quality metric, which shows a significant connection between metrics and quality):

(1) A quantitative measure of the degree to which an item possesses a given quality attribute.

(2) A function whose inputs are software data and whose output is a single numerical value that can be interpreted as the degree to which the software possesses a given quality attribute.

A similar definition of software metrics, like this in item (2), is given in reference [19].

As output metric gives statistical/numerical value and it can be the lowest value in a given time interval, the highest in the interval, and the average value of the interval. In addition, metric can be given as an absolute or percentage value. Together with metrics, we may present other statistical indicators such statistical distributions, correlation, benchmarking, trends, graphs of various types.

CLASSIFICATION OF METRICS

Software metrics can be classified in several ways depending on the criteria. According to the criteria of the software system state is divided into static and dynamic metrics. Thus, the dynamic measurement metrics collected over running software (sometimes called external metrics). Static metrics are collected through measures made by the system representation and which do not rely on software execution (internal metrics).

Management side of all metrics can be divided into control metrics and prediction metrics. Control metrics is used by management in order to control the processes related to software, examples of the

required work and the use of disk space. Control metrics can provide information about the quality of the process and the quality of the product itself. Prediction metrics extent attributes of the software product to predict the future of software quality.

In reference [1], we have given taxonomy framework with candidate metrics, which is summarized in Table 1.

Table 1. *Types group of software metrics, constructed from [1]*

Types group:	Group of metrics:
Target types	Product metrics
	Process metrics
	Hybrid metrics
Structure types	Elementary metrics
	Composite metrics
Obtainment criterion types	Objective metrics
	Subjective metrics

The same has been given more types of metrics including: design, size, complexity, reuse, productivity, quality. [1]

If we observe the software development life cycle, it is possible for each specific phase of software development methodologies to define specific software measures that will result with numeric metric. In addition to any action in the stage of software development, it is possible to define applicable metrics with the preparatory-final stages.

Object-oriented (OO) programming is tremendously different from the models of procedural, structural or functional programming. For the quality of object-oriented structure, special OO metrics are used. Object-oriented metrics respects the principles that are uniquely characteristic of object-oriented design, such as classes, objects, methods, inheritance, polymorphism, encapsulation, composition, delegations and other OO concepts.

CONSTRUCTION OF QUALITY METRICS

There are no standardized and universal applicable software metrics. Software metrics should provide control of the software development project, its maintenance, support in decision-making by software managers, monitoring and initiating corrective actions. Construction of a highly reliable software depends on the participation of the attributes of quality at every stage of the life cycle of development with emphasis on the prevention of errors, especially in the early stages. In order to measure these attributes so as to improve the quality and reliability of software, it is necessary to define metrics for each development stage (required documentation, source code, test plans and testing).

A common case is that there is an excessive number of metrics that should be reduced to a small number of simple and friendly metrics depending on the environment and aspect.

Customer requirements are specified functionality that must be included in the final software, which must be structured, complete and clear communication between the designer and the user. Two important metrics for evaluating the ambiguity of the term are a number of imprecise and general phrases (e.g. adequate, appropriate, normal, etc.) and number of optional phrases (e.g. may be optionally etc.). Incomplete terms such as "should be something added and should be specified" should also be avoided.

SOFTWARE METRICS AND BANKING

One of the most useful metric defined a long time ago (1976, Thomas J. McCabe) is cyclomatic complexity. This metric account the complexity of the whole or part of the program code by finding the number of linearly independent paths in the graph arising from the structure of the program code (Figure 1).

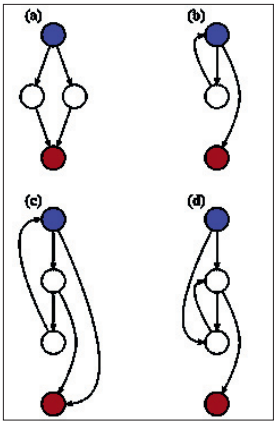

Figure 1. *Example of control flow graphs for cyclomatic complexity, adapted from [20]*

There are a number of software for the calculation of the metrics. This paper aims at banking information system of a general type of small banks with around 150,000 clients in the Republic of Srpska (en-

tity of Bosnia & Herzegovina). Aamong others, Visual Studio IDE is used for developing software solution. NDepend can be used as a general software for the calculation of the metrics. It is also used for other purposes in order to improve the targeted software. In version 6 it there 86 software metrics that cover six application areas [2]. However this software is too big for the needs of the respective banks.

Risk management is one of the most important concerns of modern banking. It is also the important work of software managers because the banks' operations are based 99% on the use of computer equipment and corresponding software. Software manager is obliged to provide all the risks that could arise during the operation and realization of software projects. Then it is necessary to adequately respond to these risks, minimizing or completely avoiding them. This is based on the statistical data after the application-specific metrics.

Development teams, or software managers, using specific metrics can identify risks, see the stage and the situation in which the current project is located and track progress during software development lifecycle. As better and a free software tool for the needs of the target bank, Code Metrics Viewer is chosen.

A set of metrics that are crystallized in the bank's projects and written in Microsoft Visual Studio 2019 environment is given below and are demonstrated by plug-in Code Metrics Viewer [3]: MI - maintainability index, CyC - cyclomatic complexity, DOI - depth of inheritance, CoC - code coupling (coupling of a program code with the code of another class), and the most simple metrics LOC - lines of code (the number of lines of program code) (Figure 2).

In the Code Metrics Viewer, we may request result of software metrics for the entire solution or project, namespace, class, method, or section.

CONCLUSION

The idea and goal of this optimized model of software metrics is to address all potential problems in a way that will offer a set of metrics and their parameters. This will allow developers to provide information on optimum values for writing quality program code, i.e. limits within which they can write their own program code, which complexity their codes could have, how to maintain index that their codes must have, depth of inheritance in the program code, and the required lines of code per writing module.

However, it was noted that this is not enough per se. It is necessary to calculated the software metrics connects to one of the methodology for software development, especially for the phases of planning and testing. As a candidate, the movement, approach and practical techniques titled DevOps is taken into consideration.

The contribution of this paper is reflected in the testing and presentation of metrics and metrics software. It is concluded that this approach greatly improves the work of software engineers and users of software produced by this approach. Details of the metrics themselves have not been released due to strict regulations on data confidentiality in the banking sector and internal rules and decisions of the bank.

Code Metrics Viewer

Analyze Solution | Compare... | Maintainability Index | Min: | Max: | Goto Next

Hierarchy	Maintainability Index	Cyclomatic Compl...	Class Coupling	Depth of Inheritance	Lines of Code
Keygen.dll	82	81	43	2	169
Keygen	84	28	23	2	70
GuidKeyGenerator	81	4	3	2	8
GenerateKey(int) : string	69	2	2		6
GuidKeyGenerator()	98	1	1		1
GuidKeyGenerator()	91	1	0		1
IKeyPairGenerator	100	1	1	0	0
IUniqueKeyGenerator	100	1	0	0	0
KeyData	94	5	0	1	5
KeyData()	100	1	0		1
PrivateKey.get() : string	98	1	0		1
PrivateKey.set(string) : void	95	1	0		1
PublicKey.get() : string	98	1	0		1
PublicKey.set(string) : void	95	1	0		1
KeyGeneratorBase	67	6	8	1	20
KeyPairGenerator	65	11	20	2	37

Figure 2. *Software metrics in Code Metrics Viewer 2015*

REFERENCES

[1] Abreu, Fernando Brito, Rogério Carapuça, *Candidate Metrics for Object-oriented Software within a Taxonomy Framework*, Journal of Systems and Software 26.1, pp. 87-96., 1994.

[2] Code metrics in NDepend version 6, http://www.ndepend.com/docs/code-metrics

[3] Code Metrics Viewer 2015, https://visualstudiogallery.msdn.microsoft.com/ee46c9de-0890-4447-910d-d2b708de71bf

[4] Drljača, Dalibor & Latinović, Branko & Starčević, Dušan. (2017). *Modelling the Process of is Auditing in the Public Administration Using UML Diagrams*. JITA - Journal of Information Technology and Applications (Banja Luka) - APEIRON. 13. 10.7251/JIT1701032D.

[5] Džakula I., Latinović B., *Reduction of Ict Security Risks Using Level Based Approach*, JITA – Journal of Information Technology and Applications Banja Luka, PanEuropien University APEIRON, Banja Luka, Republika Srpska, Bosna i Hercegovina, JITA 9(2019) 2:99-105, (UDC: 06.3:[004.738.5:316.774), (DOI: 10.7251/JIT1902099DZ), Volume 9, Number 2, Banja Luka, december 2019 (49-128), ISSN 2232-9625 (print), ISSN 2233-0194 (online), UDC 004

[6] Estdale, J., Georgiadou, E., *Applying the ISO/IEC 25010 Quality Models to Software Product*, 25th European Conference, EuroSPI 2018, Bilbao Spain, 2018, 10.1007/978-3-319-97925-0_42

[7] Fenton N, Bieman J., *Software Metrics: a Rigorous and Practical Approach*, CRC Press, 2015.

[8] Grady, Robert B., *Practical Software Metrics for Project Management and Process Improvement*, Prentice-Hall Inc., 1992.

[9] EEE Standard Glossary of Software Engineering Terminology 610.12-1990, 1990., http://ieeexplore.ieee.org/document/159342/references

[10] SO 9000 - Quality Management, 2015, http://www.iso.org/iso/home/standards/management-standards/iso_9000.htm

[11] ISO/IEC 9126-1:2001, 2001., Software Engineering -- Product Quality -- Part 1: Quality Model http://www.iso.org/iso/catalogue_detail.htm?csnumber=22749

[12] ISO/IEC TR 9126-2:2003, 2003., Software Engineering -- Product Quality -- Part 2: External Metrics, http://www.iso.org/iso/catalogue_detail.htm?csnumber=22750

[13] ISO/IEC TR 9126-3:2003, 2003., Software Engineering -- Product Quality -- Part 3: Internal Metrics, http://www.iso.org/iso/catalogue_detail.htm?csnumber=22891

[14] SO/IEC TR 9126-4:2004, 2004., Software Engineering -- Product Quality -- Part 4: Quality in Use Metrics, http://www.iso.org/iso/catalogue_detail.htm?csnumber=39752

[15] Kaner, Cem, *Software Engineering Metrics: What Do They Measure and How Do We Know* in 10th International Software Metrics Symposium, 2004.

[16] Lincke, Rüdiger, Jonas Lundberg, Welf Löwe, *Comparing Software Metrics Tools*, ISSTA '08, Proceedings of the 2008 international symposium on Software testing and analysis. ACM, 2008.

[17] Obrenović, Ž., & Starčević, D. (2006). *Adapting the Unified Software Development Process for User Interface Development*. Computer Science and Information Systems, 3(1), 33-52.

[18] Ris K., Stanković Ž., Avramović Z. Ž., *Implications of Implementation of Artificial Intelligence in the Banking Business in Relation to the Human Factor*, JITA – Journal of Information Technology and Applications Banja Luka, PanEuropien University APEIRON, Banja Luka, Republika Srpska, Bosna i Hercegovina, JITA 10(2020) 1:49-57, (UDC: 004.8:336.7]:007.52, 004.5), (DOI: 10.7251/JIT2001049R), Volume 10, Number 1, Banja Luka, June 2020 (1-68), ISSN 2232-9625 (print), ISSN 2233-0194 (online), UDC 004

[19] Schneidewind, Norman F., *Methodology for Validating Software Metrics*, IEEE Transactions on Software Engineering 18.5, pp. 410-422., 1992.

[20] Some types of control flow graphs, Wikipedia – the free encyclopedia, https://en.wikipedia.org/wiki/File:Some_types_of_control_flow_graphs.s

Submitted: December 7, 2020
Accepted: May 31, 2021

ABOUT THE AUTHORS

Dragoljub Pilipović was born on 1978 in Novi Grad (ex. Bosanski Novi). Primary and secondary education finished in Prijedor. He graduated basic and master study from Faculty of Organisational Science in Belgrade with mentor Dusan Starcevic in 2014. In 2016 successfully defended his doctoral dissertation on Faculty of Information Technology on Slobomir P University in Bijeljina.

Dejan Simeunović was born on 1981 in Gradacac. Primary and secondary education finished in Modrica. He graduated basic and master study from Faculty of Information Technology on Slobomir P University in Doboj in 2016.

FOR CITATION

Dragoljub Pilipovic, Dejan Simeunovic, The Quality of Software Metrics, *JITA – Journal of Information Technology and Applications Banja Luka*, PanEuropien University APEIRON, Banja Luka, Republika Srpska, Bosna i Hercegovina, JITA 11(2021) 1:61-65, (UDC: 519.68:681.3.06]:336.71), (DOI: 10.7251/JIT2101061P), Volume 11, Number 1, Banja Luka, June 2021 (1-68), ISSN 2232-9625 (print), ISSN 2233-0194 (online), UDC 004